The Untangler Project

by Tom Verhoeff and Christian Eggermont

We are fascinated by disentanglement puzzles. Why? Probably, at least in part, because, unlike put-together puzzles (pieces-in-box), take-apart puzzles (puzzle boxes), or inter-locking solid puzzles (burrs), there seems to be little theory to support the design and solution of disentanglement puzzles. We have been intrigued by (our lack of under-standing) how disentanglement puzzles are designed and what it takes to solve them. Nevertheless, it also appears that you can do some systematic thinking about these puzzles.

In this article we present a project to develop open-source software to assist in the solution and design of (a limited class of) disentanglement puzzles. It is a follow-up on the presentation about this topic at the Dutch Cube Day in October 2008.

For other puzzle types there is plenty of computer assistance. Computers are powerful enough to offer a good graphical user interface, and to do extensive complicated searches. So, we think the time is right to start working on disentanglement puzzles now. The project already has a presence at SourceForge:

http://untangler.wiki.sourceforge.net/

We are looking for people who are crazy about disentanglement puzzles and who would like to participate in the project. This article tells you about this effort.

Kinds of disentanglement puzzles

We believe that there is a reasonable opportunity to provide computer aided support for solving, and thereby also for designing, a subclass of disentanglement puzzles. In particular, we intend to focus on puzzles that are mainly topological, without any secrets to be discovered.

If the puzzle consists almost exclusively of (geo)metric details, such as Hanayama's *Cast Star* (Figure 1), *Cast Enigma*, and *Cast Quartet* (Figure 1), then this information needs to be modelled in the computer very accurately. But even when you can do that, the 3D motions to change the state without violating the constraints can be very difficult to discover, in particular determining whether an escape exists at all, and even harder to describe in an understandable way. Geometric disentanglement puzzles are related to the industrial field of robot motion planning, but this is not where we think we can contribute.

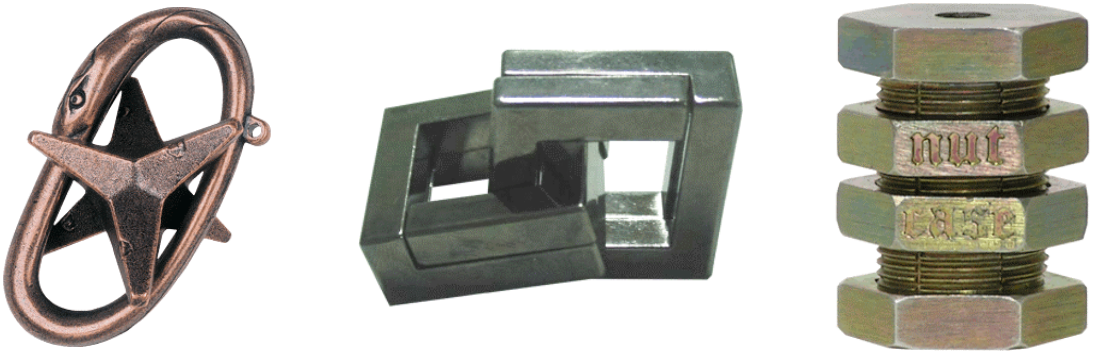

Figure 1. *Cast Star* (left), *Cast Quartet* (middle) and *Cast Nut Case* (right)

If the puzzle involves secrets (hidden information), such in Hanayama's *Cast NEWS* and *Cast Nut Case* (Figure 1), then it also seems hard to provide decent computer assistance (except maybe by consulting some database of known tricks).

We have in mind puzzles like the *Chinese Rings* (Figure 2), *Ball and Chain*, *Narrow Escape*, and *Hemi-Spheres* (Figure 2). These are mostly topological in nature. One motivation is that there exist some interesting programs to assist in the area of untying knots. For instance, in 1994 Lars Gislén wrote the program called KnotSolver. He recently did some work to make available a Java version. There have also been some recent scientific publications on particular disentanglement puzzles. See the Untangler wiki pages for details and more references.

Figure 2. *Chinese Rings* (left) and *Hemi-Spheres* (right)

The Untangler Program (a vision)
We envision a program for stand-alone use on a personal computer, with a graphical user interface plus some algebraic notation. The initial version will possibly only handle an even more limited class of topological disentanglement puzzles.

A key ingredient is how to describe such disentanglement puzzles. The following items somehow need to be captured:

- Elements that make up the puzzle: rigid parts, like rings and wireframes, hinges, ropes, chains, beads, disks, slotted beams, etc.
- Relationships between elements: attachment, embracement, etc.
- Constraints: (in)ability to slide through each other, etc.
- Initial state and final state (the objective).
- Continuous versus discrete, and metric versus topological aspects; e.g. the exact rope length (continuous/metric information) may not be so important, but whether the rope is long enough (discrete, topological information) to allow one to slide a loop over some ball could be important.

One also needs to have a pretty good idea of how to describe a solution. A solution typically consists of a feasible path in the state space, leading from the initial state to the final state, i.e. adhering to all constraints. Such a path can be described by a sequence of state changes or as a sequence of discrete intermediate states, where the user can readily infer how to fill in the actual continuous motions.

Ideally, topological disentanglement puzzles can be described completely algebraically, without resorting to pictures. A nice example of such a puzzle is *Rope-and-Rope* by Markus Goetz (see Figure 3).

Figure 3. *Rope-and-Rope* puzzle

Rope-and-Rope consists of a (wooden) ring R, (wooden) bead B, (wooden) disk D, (wooden) beam L bent at a right angle with slot S, (rope) cord C, (rope) noose N, with relationships and constraints as described in the following.

D is attached to one end of C; the other end of C is attached to L, in the middle. C goes through B. S is in one half of L, and N is attached to the other end of L. Table 1 characterises some relations between the puzzle elements of *Rope-and-Rope*.

element	R	B	D	L	S	N
R	x	over	not over	over	thru	thru
B	thru	x	not over	not over	not thru	(thru)
D	not thru	not thru	x	x	thru	thru
L	thru	not thru	x	x	x	x
S	over	not over	over	x	x	x
N	over	(over)	over	x	x	x

Table 1. Relationships between puzzle elements of *Rope-and-Rope*

The length of C is not given explicitly, but whatever is relevant about this length will be discovered while exploring the state space.

Initial state: R is around L near the middle and on the side with S; C goes through S, through N, back through S. B sits next to D. Final state: R is free.

You can imagine that a textual description of a solution starts as follows:

1. Put D back through S in the opposite direction of C attaching to D.
2. Slide R off L, in the direction of S, making R go over C six times.
   (The rope is long enough to do this.)
3. Push B through R with C. R now goes four times over C.
4. ...

On YouTube you can find movies with solutions to several disentanglement puzzles. See the Untangler wiki for links. But we will not aim for automatic generation of computer animations, although it is an interesting extension to consider.

Ingredients for the program

The program can somehow be told (through a file or GUI) what puzzle to solve. It then applies generic techniques to construct and explore the state space. If it finds a solution, the program describes it in some algebraic notation. At a later stage, visualisations of the state, and animations of state changes might be added.

Several techniques come to mind to search for and find solutions. In general, this involves a systematic exploration of the state space. Typical techniques that are useful: backtracking, working backwards (starting in the final state and trying to move back toward the initial state), and relaxation, where one first drops some constraints and attempts to solve a simplified puzzle first.

It may also be useful to have the program indicate that is impossible to solve the puzzle, and provide evidence for the insolvability.

It may be necessary to incorporate user interaction, when the state space cannot easily be generated automatically. In that case, the program could ask the user to confirm whether a particular move is actually feasible. If it is, then that part of the state space is included, and if not, it is pruned. For the *Rope-and-Rope* puzzle, this is may be the simplest way to deal with the constraint imposed by the length of the cord.

Call for help

We do not have a lot of time to spend on this project. But still we would like to make some progress. We need help with the following:

- Discovering and describing the requirements for the software in more detail.
- Searching for relevant theory and literature on this topic.
- Inventing appropriate concepts and notations and further develop relevant theory.
- Setting up a generic software architecture.
- Implementing prototypes for some small-scale experiments.
- Providing feedback on what has been done.

If you are enthusiastic about this, please drop us an email.

Tom Verhoeff: T.Verhoeff@tue.nl
Christian Eggermont: C.E.J.Eggermont@tue.nl